On the Learning of Deep Local Features for Robust Face Spoofing Detection

Gustavo Botelho de Souza[1], João Paulo Papa[2] and Aparecido Nilceu Marana[2]
[1]UFSCar - Federal University of São Carlos. Rod. Washington Luís, Km 235. São Carlos (SP), Brazil. 13565-905.
[2]UNESP - São Paulo State University. Av. Eng. Luiz Edmundo Carrijo Coube, 14-01. Bauru (SP), Brazil. 17033-360.
E-mail: gustavo.botelho@gmail.com, {papa, nilceu}@fc.unesp.br

***Abstract*—Biometrics emerged as a robust solution for security systems. However, given the dissemination of biometric applications, criminals are developing techniques to circumvent them by simulating physical or behavioral traits of legal users (spoofing attacks). Despite face being a promising characteristic due to its universality, acceptability and presence of cameras almost everywhere, face recognition systems are extremely vulnerable to such frauds since they can be easily fooled with common printed facial photographs. State-of-the-art approaches, based on Convolutional Neural Networks (CNNs), present good results in face spoofing detection. However, these methods do not consider the importance of learning deep local features from each facial region, even though it is known from face recognition that each facial region presents different visual aspects, which can also be exploited for face spoofing detection. In this work we propose a novel CNN architecture trained in two steps for such task. Initially, each part of the neural network learns features from a given facial region. Afterwards, the whole model is fine-tuned on the whole facial images. Results show that such pre-training step allows the CNN to learn different local spoofing cues, improving the performance and the convergence speed of the final model, outperforming the state-of-the-art approaches.**

I. Introduction

Biometric systems are increasingly common in our everyday activities [1]. People recognition through their own physical, physiological or behavioral traits inhibits most of the frauds often committed in security systems based on knowledge (passwords) or tokens (cards, keys, etc.). However, nowadays criminals are already developing techniques to accurately simulate the biometric characteristics of valid users, such as face, fingerprint and iris, to gain access to places or systems, process known as spoofing attack [2], [3]. In this context, robust countermeasure techniques must be developed and integrated into the traditional biometric applications in order to prevent such frauds. Despite face being a promising trait due to its convenience for users, universality and acceptability, traditional face recognition systems can be easily fooled with common printed facial photographs [2], which nowadays can be obtained by criminals on the worldwide network, especially due to the dissemination of social medias and networks.

Spatial image information is extremely important in tasks involving faces, such as face detection [4] and face recognition [5], [6]. The different visual patterns of each facial region encode rich and discriminative information necessary to distinguish a face from other objects, and also from other faces. Regarding face spoofing detection, some works based on handcrafted features have mentioned that different spoofing cues can be extracted from different facial regions [7], [8].

Recently, deep learning architectures have emerged as good alternatives for solving complex problems and have reached state-of-the-art results in many tasks due to their great power of abstraction and robustness, working with high-level features, self-learned from the training data [9], [10]. Among the proposed deep learning architectures, Convolutional Neural Networks (CNN) [11] have appeared as one of the most important classes of deep neural networks able to deal with digital images with great performances.

Some CNN based state-of-the-art methods were recently proposed for face spoofing detection [12]–[15]. However, none of them take into account the different visual aspects of each facial region and, consequently, the different local spoofing cues that could be learned by the neural networks to improve their performances. All proposed methods work on whole faces, in a holistic way, or with random and small patches, i.e., they train the neural networks with samples extracted from random regions of the faces, all together. This can degrade the performance of the training algorithm since the backpropagation method can be distracted by the different visual information extracted from random regions of the face, instead of learning the real differences between real and fake faces in each facial region, with similar visual aspects, differing only by spoofing cues.

In this context, we propose a novel CNN architecture trained in two steps for a better performance in face spoofing detection: (i) the local pre-training phase, in which each part of the model is trained on each main facial region, learning deep local features for attack detection and initializing the whole model in a great position in the search space (the network learns to detect multiple and different spoofing cues from all the facial regions); (ii) the global fine tuning phase, in which the whole model is fine-tuned based on the weights learned independently by its parts and on whole real and fake facial images, in order to improve the model generalization. Results obtained on two major datasets used for the evaluation of face spoofing detection techniques, Replay-Attack [8] and CASIA FASD (Face Antispoofing Database) [16], show that the pre-training step on local and fixed regions of the faces improves the performance of the final model and its convergence speed. The proposed approach outperformed the state-of-the-art methods while working with an efficient CNN architecture.

II. Technical Background

In this section we briefly present some concepts regarding the importance of spatial information and differences of the facial regions for face detection, face recognition and face spoofing detection, as well as some related works.

A. Facial Regions and Spatial Information

The spatial relationship between the facial elements and regions in the images encodes rich information that can be used to distinguish a face from the background, from other objects or even from other faces [4], [5]. The first works on automated face detection and recognition already used such kind of information, presenting good results and efficiency.

Regarding face detection, the early work of Viola and Jones [4] used Haar-like features to detect the presence of faces in digital images. In short, they apply, to each area of a given image, a cascade classifier which verifies, hierarchically, whether all main facial features are present in that area. The Haar-like features are designed to capture typical differences existing in neighboring regions of human faces. Fig. 1 shows two Haar-like features and their correspondence to the regions of human faces. The black rectangles indicate that darker regions are expected, while white rectangles indicate that brighter regions are expected in a certain area. The feature showed in the middle focus on darker and brighter regions corresponding to the eyes (especially due to eyebrows) and cheeks, respectively. The feature on the right searches for the contrast of the nose and eyes in human faces.

Fig. 1. Left: face detected based on the Haar-like features used by Viola and Jones [4]. Center and right: examples of Haar-like features. Images from the OpenCV documentation [17].

Based on the work of Viola and Jones [4], which allowed automated face detection in reasonable time for real applications, many works were later proposed that also explored the contrasts existing in neighboring regions of the face [18]–[20].

In the context of face recognition, the first effective method for real scenarios was proposed by Turk and Pentland [5], based on the Principal Component Analysis (PCA) [21], which can be used to find the most discriminative eigenvectors that best describe the variance of the set of data under analysis (facial images, in this case) and reduce the dimensionality of the problem. Given the similarity of such eigenvectors (when represented as 2D images) to facial images, Turk and Pentland called them eigenfaces [5], [22]. It is possible to identify the facial elements and regions (and their spatial relationship) in the eigenfaces, indicating that this kind of information is important to differentiate faces from different people.

Works based on other transformations for reducing the dimensionality of the facial images space, such as the ones based on the Linear Discriminat Analysis (LDA) [23], also usually obtain, as the "basis" of the new coordinate systems, vectors that ensemble human faces when viewed as 2D images, with the different facial regions in them being noticeable. The CNN based architectures for face recognition, which self-learn the most discriminative features for face representation from the training datasets, also capture the spatial information and relationships between facial elements and regions, presenting connection weights between neurons that act as edge and facial elements detectors (eyes, nose, etc.) [24].

Researches in Psychology show that human beings have an extreme ability to detect faces, more accurately and much faster than any other object, and also highlighted the importance of spatial information and the positioning of each facial region and element for face detection and recognition [25], [26]. In [26], for instance, the authors found that the time required by a group of people to identify a visual stimulus as a face was shorter when normal faces were presented than when jumbled faces, i.e., faces with parts out of place (the mouth region above the eyes, etc.), were presented.

Despite all this, to the best of our knowledge, no work has investigated the usage of deep local features, learned from each facial region (with its particular visual aspect), to improve the performance of the state-of-the-art deep learning architectures for face spoofing detection, our main goal.

B. Face Spoofing Detection

According to Ratha, Connell and Bolle [3], as in any other security system, there are many ways to attack a biometric system. In short, the attacks to biometric applications can be divided in two groups: direct and indirect attacks. In the direct attacks (spoofing attacks), criminals generate synthetic samples of biometric traits of legal users, such as photographs (face simulation), gelatin fingers (fingerprint simulation), contact lenses (iris simulation), among others, to obtain access to places or systems. Criminals try to fool the capture sensor with such samples, the most vulnerable point of the biometric recognition system [3].

In the indirect attacks, criminals, after investigating the inner working of the system and based on some fragility, act by modifying the algorithms used to match templates or internal messages exchanged by the system modules [3]. Fig. 2 shows the main points of attack of a biometric system. It is important to know, however, that the vast majority of attacks on biometric applications are direct, due to the simplicity for attackers who do not need to investigate the inner working of the system.

Among the main biometric traits, as said, face is a promising one especially due to its convenience, low cost of acquisition, universality and acceptability by users [1], being very suitable to a wide variety of environments, including mobile ones. However, despite all these advantages, face recognition systems are the ones that most suffer from spoofing attacks since they can be easily fooled even with common printed photographs [2]. Fig. 3 shows some real and fake faces from the Replay-Attack [8] dataset. As one can observe, it is very difficult to distinguish between real and fake faces.

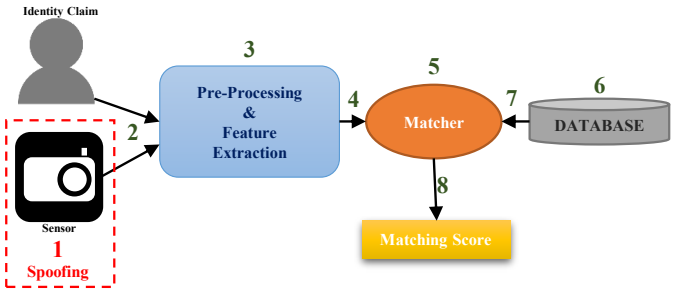

Fig. 2. Points of attack in a traditional biometric system. The spoofing attacks occur in point "1", i.e., by fooling the sensor (presentation of fake traits) [27].

Regarding face spoofing attacks, these can be performed by presenting to the camera of a biometric system a static face image (printed, digital image displayed on a mobile device, or a 3D mask) or a dynamic set of face images (videos recorded from the faces of legal users displayed on mobile devices) [2]. As one can observe in Fig. 3, different spoofing cues can be analyzed in each facial region, such as shadows (more common in real faces, especially in their outer regions, than in 2D fake faces).

C. Related Works

Face spoofing detection methods have been proposed in literature in the last years. Regarding the approaches that work with handcrafted features, most of them focus on detecting spoofing artifacts and image quality distortions in order to identify fake faces. Some of them, such as [28], extract color features, based on the assumption that, when recaptured by the cameras of the biometric systems, fake faces present distortions in colors, reflectance, etc., due to the properties of the materials they are made with. In [28], the authors argue that fake faces tend to present darker colors and different contrasts, as well as more low-frequency areas than real faces. They use such information to extract features for face classification.

Other works extract texture features based on the LBP (Local Binary Patterns) [29] descriptor and its variations, to characterize real and fake faces, presenting good results [30]–[32]. In [30], the authors extract specific features from each facial region and combine them into a final feature vector in the end of the process for classification, improving significantly the results of the method compared to when working with features from the whole face. Some of these works also mentioned that the best features were extracted from specific facial regions, especially the central one and from the forehead area [30].

Fig. 3. Images from real (first row) and fake faces (second row) from the Replay-Attack [8] dataset.

Among the approaches for face spoofing detection which use deep learning architectures, more specifically Convolutional Neural Networks (CNN) [11], since for this task they obtained the state-of-the-art results, to the best of our knowledge, all of them work on whole faces, learning global spoofing cues, or on random and small patches extracted from the faces, not focusing on the learning of local spoofing cues from each facial region. In [12], for instance, the authors apply a transfer learning algorithm in order to adapt the VGG-Face [24] model, a benchmark CNN for face recognition trained on 2.6 million facial images from $2,622$ people, for spoofing detection, obtaining good results given the similar domains of the problems. In [33], the authors also apply a similar transfer learning algorithm on VGG-Face [24] model using it for feature extraction, without modifying the original model, focusing on efficiency. In [34], a more time consuming algorithm for transfer learning is applied to the VGG-Face [24] neural network, in which layers of the original CNN are updated for the spoofing detection task, obtaining great results, but also making the process more expensive and requiring more processing power (advanced GPUs) and time.

All these aforementioned works based on the VGG-Face [24] consider whole facial images as input. Other important works in the literature such as [15], [35] also extract global deep spoofing cues from the faces based on other architectures. In [14], the authors propose a CNN model and integrate it with a Long-Short Term Memory (LSTM) [36] neural network for learning temporal holistic features from the faces in sequences of images (videos), also obtaining a good performance.

In [13], the authors explore random patches for face spoofing detection. They use such approach especially for augmenting the dataset but present the patches all together (from random and different parts of the faces) to train their CNN architecture. Despite the good results, given the different visual patterns of each facial region, the neural network can be distracted and base its learning for spoofing detection mainly on the structural information of the faces, much more evident in the images, not focusing on the spoofing cues themselves. In other words, the backpropagation algorithm can be more influenced by the structural aspects of the facial elements (e.g. presence or absence, size, shape, etc. of the eyes) in a given patch, than by the subtle spoofing cues in it.

Another well-known patch based approach for face spoofing detection, presented in [7], works with small and not fixed patches (regions) from the faces to train traditional classification models. In each face, given an extensive analysis based on several metrics, they select the best patches to represent the whole facial image in order to classify it as real or fake. They use many metrics to determine which patches should be selected to represent the face, which are obtained from

different regions of the faces for each sample, also degrading the performance of the method in learning spoofing cues.

Despite the lack of attention to deep local features regarding face spoofing detection, Krizhevsky [37] demonstrated, on other image classification tasks, that the use of local (and fixed) regions of the images (visual local information), in an initial training step of the deep learning model, tends to improve its performance, also avoiding getting stuck in local minima in the hyperparameter search space. Ba et al. [38] also suggested the use of facial patches for initializing deep models applied to face recognition based on studies in Neuroscience. Another work [39] uses this initial training step based on fixed image patches for improving vehicle classification in images.

III. Proposed Approach

In this work, we propose a novel CNN architecture for face spoofing detection, which we called lsCNN (Locally Specialized CNN), with a novel training algorithm for a more effective learning of deep local spoofing features, based on two steps: (i) the local pre-training phase, in which each part of the model is trained on each main facial region (predefined and fixed), learning deep local features for attack detection and allowing to initialize the whole model in a better position in the search space; and (ii) the global fine tuning phase, in which the whole model is fine-tuned based on the weights learned independently by its parts on the facial regions, in order to improve its generalization.

A. lsCNN Architecture

Basically, the lsCNN presents 4 convolutional and pooling layers ($Conv1/Pool1$ to $Conv4/Pool4$) at the bottom, with each convolutional layer being immediately followed by a batch normalization, scale and signal rectification (ReLU - Rectified Linear Unit) layers. The batch normalization and scale layers serve to normalize the output feature maps obtained in the convolutional layers, improving learning [40]. The rectification function, in each neuron, acts as activation function, eliminating negative values in the resultant feature maps and also accelerating training. At the top of the network is a fully-connected layer ($FC1$), also followed by a batch normalization, scale and ReLU layers, as well as a dropout one ($Drop1$). Finally, there is a softmax layer with two neurons in order to classify the faces as being real or fake. Tab. I presents the lsCNN architecture in terms of its layers, i.e., size of kernels, strides, sizes of input and output feature maps.

As shown in Tab. I, lsCNN expects 3-channels facial images in RGB color space as input. Although other color spaces allow dealing more accurately with illumination issues, in order to approximate the model to the inner working of human eyes (which capture only red, green and blue waves of light) and their perception in natural conditions, as well as by the fact that most digital cameras capture images in RGB mode, we opted for this image representation over other color models.

B. Local Pre-training

Similar to [37], in order to initialize the whole lsCNN model in a better position in the search space and make it specialized

TABLE I
Architecture of the proposed lsCNN. The inputs of lsCNN are RGB (3 channels) facial images with 96×96 pixels: $3 \times (96 \times 96)$ maps.

Layer	Kernel Size	Stride	Input Maps	Output Maps
Conv1	3×3	1	$3 \times (96 \times 96)$	$27 \times (94 \times 94)$
Pool1	2×2	2	$27 \times (94 \times 94)$	$27 \times (47 \times 47)$
Conv2	3×3	1	$27 \times (47 \times 47)$	$36 \times (45 \times 45)$
Pool2	2×2	2	$36 \times (45 \times 45)$	$36 \times (23 \times 23)$
Conv3	3×3	1	$36 \times (23 \times 23)$	$45 \times (21 \times 21)$
Pool3	2×2	2	$45 \times (21 \times 21)$	$45 \times (11 \times 11)$
Conv4	3×3	1	$45 \times (11 \times 11)$	$54 \times (9 \times 9)$
Pool4	2×2	2	$54 \times (9 \times 9)$	$54 \times (5 \times 5)$
FC1	—	—	$54 \times (5 \times 5)$	$1 \times (450)$
Drop1	—	—	—	—
Softmax	—	—	$1 \times (450)$	$1 \times (2)$

in deep local spoofing features from each region of the faces, we split each training face into 9 main regions (patches), as shown in Fig. 4, regions also adopted for face recognition [1].

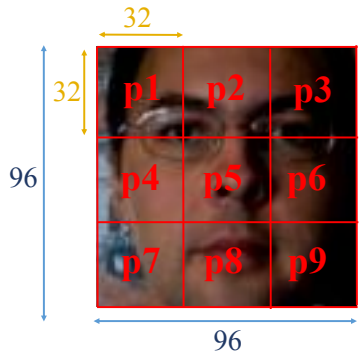

Fig. 4. A face image (96×96 pixels) from the Replay-Attack dataset [8] split into 9 fixed patches (non-overlapping regions of 32×32 pixels).

After this, we also split the lsCNN architecture into 9 independent smaller CNNs, called PatchNets for simplicity, presenting, each of them, a ninth of the size of the original model, and being trained on each of the 9 main facial regions considered from the faces, from $p1$ to $p9$. Each PatchNet has as input RGB patches with 32×32 pixels from a respective region of the training faces. Tab. II shows the architecture of each PatchNet and Fig. 5 illustrates the training process of the 9 instances of this smaller neural network on the facial regions of a given image. As one can observe, on the top of each PatchNet are 2 softmax neurons since they are trained to classify their respective patches as being real or fake.

C. Global Fine Tuning

After training the 9 smaller neural networks in their respective facial regions, their weights and biases are used to initialize the parts of the whole lsCNN for a fine tuning step of such larger model on the whole training facial images, in order to improve its generalization.

As shown in Fig. 6, each smaller network initializes the weights of the connections and biases of a partition (a ninth) of the lsCNN model, from the left (top) to the right (bottom)

TABLE II
ARCHITECTURE OF EACH SMALLER CNN (PATCHNET), PART OF THE LSCNN, TRAINED ON EACH FACIAL REGION, FROM $p1$ TO $p9$ (FIXED PATCHES WITH 32×32 PIXELS, ALSO IN RGB COLOR SPACE).

Layer	Kernel Size	Stride	Input Maps	Output Maps
Conv1	3×3	1	$3 \times (32 \times 32)$	$3 \times (30 \times 30)$
Pool1	2×2	2	$3 \times (30 \times 30)$	$3 \times (15 \times 15)$
Conv2	3×3	1	$3 \times (15 \times 15)$	$4 \times (13 \times 13)$
Pool2	2×2	2	$4 \times (13 \times 13)$	$4 \times (7 \times 7)$
Conv3	3×3	1	$4 \times (7 \times 7)$	$5 \times (5 \times 5)$
Pool3	2×2	2	$5 \times (5 \times 5)$	$5 \times (3 \times 3)$
Conv4	3×3	1	$5 \times (3 \times 3)$	$6 \times (1 \times 1)$
Pool4	2×2	2	$6 \times (1 \times 1)$	$6 \times (1 \times 1)$
FC1	—	—	$6 \times (1 \times 1)$	$1 \times (50)$
Drop1	—	—	—	—
Softmax	—	—	$1 \times (50)$	$1 \times (2)$

side of the lsCNN model. The weights of the first PatchNet, for example, initialize the connections between the most left neurons of the lsCNN model, responsible for first feature maps ($FM1$ to $FM3$ in the case of the first network layer), and so on (similarly to [37]). The connections of lsCNN between neurons from different parts of it are zero-initialized.

The weights of the two fully-connected layers on top are randomly initialized from a normal distribution in order to improve the generalization of model even more, as in [37]. Their biases are zero-initialized. In Fig. 6, for simplicity, in each partition of lsCNN, only the connections from a neuron in a given feature map to the neurons of the previous layer are shown, as well as the connections of the selected neurons in the first part of lsCNN to their receptive fields in the other parts of such whole model. However, the lsCNN has all the connections of a traditional CNN.

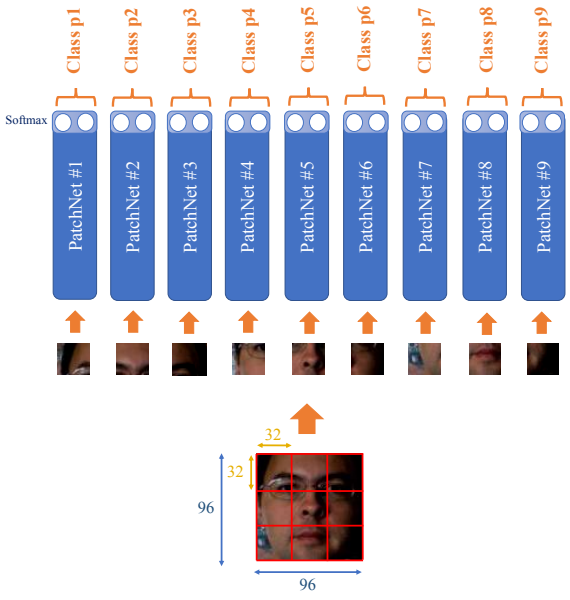

Fig. 5. Illustration of the local pre-training process of lsCNN. Given a facial image, it is split into its 9 main regions, from $p1$ to $p9$, and 9 instances of the smaller CNN architecture (PatchNet) are trained on each of them.

After the initialization, the same training facial images (which were split into patches in the former step) are used to fine tune the weights of the whole lsCNN model, also allowing it to detect some global or more generic features from whole faces, which were not learned locally in the pre-training step.

IV. EXPERIMENTS, RESULTS AND DISCUSSION

We evaluated the proposed lsCNN architecture on three important face spoofing detection databases: (i) NUAA Imposter Database [41]; (ii) Replay-Attack [8] dataset; and (iii) CASIA FASD (Face Antispoofing Database) [16]. Subsecs. IV-A and IV-B describe the experiments and the great results obtained, as well as some discussion.

A. NUAA Imposter Database

NUAA Photograph Imposter Database [41] contains grayscale facial photographs (already cropped) obtained from real and fake faces: $3,491$ images for training ($1,743$ from real faces and $1,748$ from printed facial photographs) and $9,123$ images for testing ($3,362$ from real faces and $5,761$ from printed facial images). We performed an initial experiment on this small dataset and, for this, we had to reduced the depth of the lsCNN model, eliminating the third and fourth convolutional and pooling layers due to the small size of the input faces (64×64 pixels - input patches with only 21×21 pixels). Given this reduction in depth, for this experiment we augmented the width of the original lsCNN: the first and second convolutional layers presented 90 and 135 output feature maps, respectively. The fully-connected layer presented $1,350$ neurons and, following [11], 5×5 kernels (with stride of 2 pixels) were used in the convolutions, given the formed shallow architecture. The first convolutional layer of the lsCNN and of the PatchNets had as input, respectively, $1 \times (64 \times 64)$ and $1 \times (21 \times 21)$ sized feature maps (by working with grayscale images).

The whole lsCNN model was also divided into 9 parts and we initialized all weights of the PatchNets based on random values from a zero-mean normal distribution (with standard deviation of 0.0001), and normalized the input facial images (before splitting them) by subtracting the mean value of the training set and dividing the values of the pixels by 128, in order to ensure that most of them would belong to the interval $[-1; 1]$. The biases of the neurons were all zero-initialized.

As optimizer, we used the Adam method [42], with the following parameters: 64 training images per batch, base learning rate of 0.0001, first momentum of 0.9 and second momentum of 0.999. We trained the 9 PatchNets by $2,000$ iterations using the Caffe framework [43], initialized the whole lsCNN model with their learned weights and biases, and trained the whole CNN for $2,000$ iterations on the whole training faces. For performance comparison, we also assessed a CNN with the same architecture of lsCNN, but traditionally trained, i.e., by initializing all its weights with random values extracted from a normal distribution with zero-mean and standard deviation of 0.0001 (biases zero-initialized) and training it on the whole faces also by $2,000$ iterations (its convergence point).

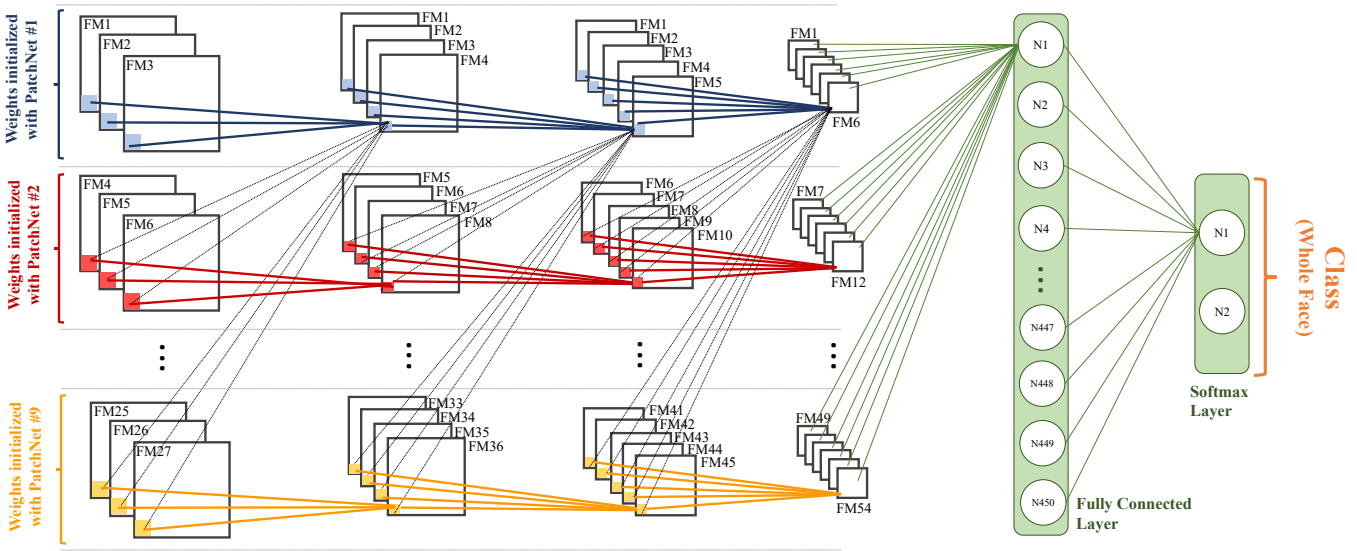

Fig. 6. Initialization of the lsCNN model based on the weights of the 9 PatchNets. The thicker colored lines represent 3×3 connections and are initialized with the weights learned by each PatchNet. The first PatchNet, for example, initializes the weights between the first neurons (first feature maps - FM) in the layers of lsCNN. The thin black dotted lines also indicate 3×3 connections zero-initialized and the green thin ones are initialized with random values from a normal distribution (zero-mean and standard deviation of 0.01, by default). The thin gray lines are just for a better visualization of the initialization process.

The goal of this initial experiment was to at first verify the improvement in performance of lsCNN compared with the traditionally trained CNN, given the same amount of training iterations (both $2,000$ iterations). Fig. 7 shows the ROC (Receiver Operating Characteristics) curves of lsCNN and the CNN traditionally trained on whole faces, learning global features. As one can observe, the proposed approach presented a much better ROC curve than the traditionally trained CNN. Regarding the Equal Error Rate (EER), the lsCNN and the traditionally trained CNN obtained, respectively, 14.10% and 23.11%. That is, our proposed approach was again much better than the traditionally trained CNN.

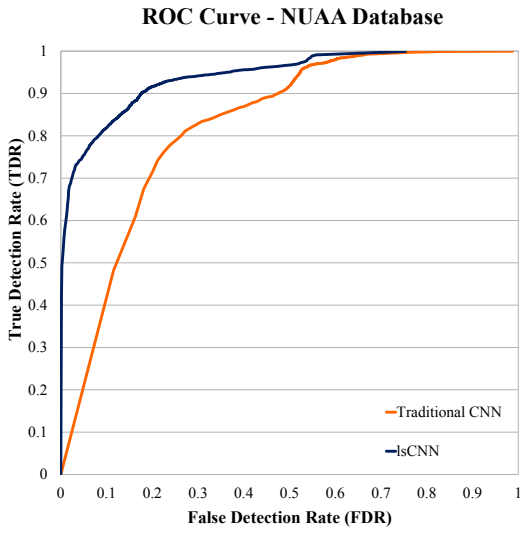

Fig. 7. ROC curves of the lsCNN and of a CNN with the same architecture, but traditionally trained, i.e., on the whole faces of NUAA [41], without a local pre-training step. The higher the curve, the better.

B. Replay-Attack and CASIA Databases

In order to allow a more robust analysis of lsCNN, we performed larger experiments on the Replay-Attack [8] and CASIA [16] databases. The Replay-Attack dataset contains 360 videos for training (60 videos of real faces and 300 videos of fake faces), 360 videos for validation in order to calibrate the threshold of the system used to determine whether a given facial image (extracted from a video frame) is real or fake, and a test set of videos with 80 videos of real faces and 400 videos of fake faces. The CASIA [16] dataset presents videos of 50 subjects, 12 videos per subject being 3 of real faces and 9 of fake faces. The dataset is divided in training set (20 subjects, 240 videos) and test set (30 subjects, 360 videos). There is no validation set explicitly defined for this database.

We detected and cropped the faces in the frames of the videos in both datasets using the robust MTCNN [44] deep neural network, for an accurate face segmentation. Based on the eyes' landmarks of a face, returned as output by MTCNN, we applied a scale transformation on the respective image in order to normalize the distance between both eyes to 60 pixels (using the MATLAB algorithm based on interpolation and on the values of the nearest pixels). After detecting and normalizing the face in each frame, we cropped it based on the eyes and capturing the whole facial region (both ears, forehead and chin), with a fixed size of 96×96 pixels in RGB color space. Some cropped faces from the Replay-Attack dataset are shown in Fig. 3. In the experiments on both datasets, in order to classify a video, we considered a majority of votes scheme of the faces in its frames. Frames with no face detected by the

MTCNN architecture were discarded.

Unlike the experiment with the NUAA dataset, in the experiments with the Replay-Attack and CASIA datasets, we considered the original architecture of lsCNN given the larger facial images obtained. After cropping the faces of all frames of all training videos, an augmentation process on both datasets was performed. In each of them, initially and for each facial image, we generated two new versions of it by increasing or decreasing the values of the R, G, and B channels by 50. This was done in order to force the network to not rely on brightness for spoofing detection (we did not apply techniques for attenuating the shadows on the faces since they are important to distinguish real faces from 2D fake faces).

For each of the three versions of each original training facial image, we also applied noise or blur transformations in three levels each (with low magnitudes to not affect the images much), in order to make the neural network also learn smoother features and not rely much on noise. Again we used the MATLAB toolbox for applying blur and Gaussian noise to the images. The blur operation was applied in three levels (using a 2×2 Gaussian filter with standard deviations of 0.1, 0.5 and 1.0), as well as the Gaussian noise (with standard deviations of 0.0005, 0.00075 and 0.001). Such transformations were applied isolatedly, so we obtained, for each of the three initial images from a given face, 6 representations of it. In this sense we augmented our dataset 19 times (original images and $3 \times 6 = 18$ transformed images).

For the Replay-Attack dataset we obtained $1,766,031$ training facial images, and for the CASIA dataset, $852,568$ images. Again, we initialized all weights of the smaller PacthNets based on random values from a zero-mean normal distribution (standard deviation of 0.0001) and normalized each channel of the input facial images by subtracting the mean value of it and diving all the image values by 128 (before splitting them), in order to ensure that most of them would belong to the interval $[-1; 1]$. The biases of the neurons were all zero-initialized. As optimizer, we also used the Adam [42] method in both cases, with the same following parameters: 64 training images per batch, base learning rate of 0.0001, first momentum of 0.9 and second momentum of 0.999.

In both experiments, we trained the 9 smaller PatchNets for $5,000$ iterations on the facial patches using the Caffe framework [43] and initialized the whole lsCNN model. Then we fine-tuned it over $100,000$ iterations. For the Replay-Attack dataset, the best model was obtained (considering results on the validation set of videos) on iteration $53,600$. For the CNN with the same architecture, traditionally initialized with random values extracted from a normal distribution with zero-mean and standard deviation of 0.0001 (biases also zero-initialized) and trained on the whole faces, the best model was obtained only on iteration $74,200$ (much later). The results of the proposed approach and of state-of-the-art methods are presented in Tab. III. For simplicity, we denoted the traditionally trained CNN with the same architecture of lsCNN as "lsCNN Traditionally Trained".

As one can observe, besides obtaining the best EER, lsCNN presented a great HTER, much better than expensive methods, such as [34], which work with extremely complex and large CNNs, such as VGG-Face [24]. Despite obtaining a worse HTER result than the traditionally trained neural network, lsCNN obtained the presented results much faster (in a much earlier iteration of the training), as mentioned.

TABLE III
RESULTS ON REPLAY-ATTACK [8] DATASET: EQUAL ERROR RATE (EER) ON THE VALIDATION DATASET AND HALF-TOTAL ERROR RATE (HTER) ON THE TEST SET. BEST VALUES ARE HIGHLIGHTED.

Method	EER	HTER
Efficient Fine-Tuned VGG-Face [33]	—	16.62
Patch Based Handcrafted Approach [7]	—	5.0
Whole Fine-Tuned VGG-Face [34]	—	**1.20**
Fine-Tuned VGG Face [12]	8.40	4.30
Li et al. [12]	2.90	6.10
Random Patches Based CNN [13]	2.50	**1.25**
Boulkenafet et al. [45]	0.40	2.90
lsCNN Traditionally Trained	**0.33**	1.75
lsCNN	**0.33**	2.50

Regarding the CASIA experiment, the best model for lsCNN was obtained on iteration $9,800$, while the best model for the traditionally trained CNN was obtained on iteration $80,900$. In order to compare the performances of such methods with state-of-the-art approaches, we measured the EER, since this dataset presents a predefined test dataset. Tab. IV shows the results.

TABLE IV
RESULTS IN THE CASIA [16] DATASET OF THE PROPOSED NETWORK ARCHITECTURE (LSCNN) AND OTHER STATE-OF-THE-ART METHODS. THE BEST VALUES ARE HIGHLIGHTED.

Method	EER
Fine-tuned VGG-Face [12]	5.20
LSTM-CNN [14]	5.17
Yang et al. [15]	4.92
Patch Based Handcrafted Approach [7]	4.65
Li et al. [12]	4.50
Random Patches Based CNN [13]	**4.44**
lsCNN Traditionally Trained	**4.44**
lsCNN	**4.44**

As one can observe, lsCNN obtained the best EER on the CASIA dataset, as well as the traditionally trained CNN and the work of [13], better than approaches that require complex and expensive architectures. Besides, when compared with the traditionally trained CNN, lsCNN training was much faster (lsCNN obtained its best performance on iteration $9,800$ against iteration $80,900$ for the lsCNN architecture traditionally trained).

V. CONCLUSION

Face spoofing detection is a critical task nowadays, given the widespread usage of face recognition systems and the development by criminals of attack techniques to simulate faces of legal users. Traditional face recognition systems can be easily circumvented with common printed facial photographs, available, nowadays, in social medias and networks.

Despite the fact that face detection and recognition methods take into account the different regions of human face for such tasks, to the best of our knowledge, no technique used deep local spoofing cues for attack detection so far, as we propose. Experimental results show a high increase in the performance of the proposed CNN architecture, lsCNN, when initialized based on a local pre-training step (on the main facial regions). The lsCNN obtained state-of-the-art results on the evaluated datasets with a quite compact model, also being much more efficient than benchmark CNNs, such as VGG-Face, which is highly used for attack detection through transfer learning.

The proposed training approach can also be applied for training other CNN models, including larger architectures, in order to improve their performances in spoofing detection as well as their efficiency during learning even more.

ACKNOWLEDGEMENTS

The authors are grateful to FAPESP (grants #2014/12236-1, #2017/05522-6 and #2016/19403-6), CAPES (grant #88881.132647/2016-01), to Dr. Anil K. Jain for the doctoral exchange period, to NVIDIA, and to Banco do Brasil.

REFERENCES

[1] A. Jain, A. Ross, and K. Nandakumar, *Introduction to Biometrics*. United States: Springer, 2011.

[2] K. Patel, H. Han, and A. Jain, "Secure face unlock: spoof detection on smartphones," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2268-2283, 2016.

[3] N. Ratha, J. Connell, and R. Bolle, "An analysis of minutiae matching strength," in *Proc. of International Conference on Audio- and Video-Based Biometric Person Authentication*, pp. 223-228, 2001.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[5] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 1991.

[6] G. Chiachia, A. Falcão, N. Pinto, A. Rocha, and D. Cox, "Learning person-specific representations from faces in the wild," *IEEE Trans. on Information Forensics and Security*, vol. 9, no. 12, pp. 2089-2099, 2014.

[7] Z. Akhtar and G. Foresti, "Face spoof attack recognition using discriminative image patches," *Journal of Electr. and Comp. Engineering*, 2016.

[8] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of Local Binary Patterns in face anti-spoofing," in *Proc. of International Conference of Biometrics Special Interest Group*, 2012.

[9] Y. Bengio, "Deep learning of representations: looking forward," *Statistical Language and Speech Processing*, vol. 7978, pp. 1-37, 2013.

[10] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *ArXiv preprint arXiv:1605.07678v4*, 2017.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[12] L. Li, X. Feng, Z. Boulkenafet, Z. Xia, M. Li, and A. Hadid, "An original face anti-spoofing approach using partial convolutional neural network," in *Proc. of Interntional Conference on Image Processing - Theory, Tools and Applications*, pp. 1-6, 2016

[13] Y. Atoum, Y. Liu, A. Jourabloo, and X. Liu, "Face anti-spoofing using patch and depth-based CNNs," in *Proc. of International Joint Conference on Biometrics*, 2017.

[14] Z. Xu, S. Li, and W. Deng, "Learning temporal features using LSTM-CNN architecture for face anti-spoofing," in *Proc. of Asian Conference on Pattern Recognition*, pp. 141-145, 2015.

[15] J. Yang, Z. Lei, and S. Li, "Learn convolutional neural network for face anti-spoofing," *CoRR, abs/1408.5601*, 2014.

[16] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Li, "A face antispoofing database with diverse attacks," in *Proc. of International Conference on Biometrics*, 2012.

[17] G. Bradski, "The OpenCV library," *Journal of Software Tools*, 2000.

[18] T. Mita, T. Kaneko, and O. Hori, "Joint Haar-like features for face detection," in *Proc. of IEEE Intl. Conf. on Computer Vision*, vol. 1, 2005.

[19] M. Mathias, R. Benenson, M. Pedersoli, and L. Gool, "Face detection without bells and whistles," in *Proc. of European Conference on Computer Vision*, pp. 720-735, 2014.

[20] S. Ma and L. Bai, "A face detection algorithm based on Adaboost and new Haar-Like feature," in *Proc. of IEEE International Conference on Software Engineering and Service Science*, 2016.

[21] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559-572, 1901.

[22] M. Dusenberry, "On eigenfaces: creating ghost-like images from a set of faces," 2015. Available at: *https://mikedusenberry.com/on-eigenfaces*

[23] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179-188, 1936.

[24] O. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. of British Machine Vision Conference*, 2015.

[25] M. Lewis and H. Ellis, "How we detect a face: a survey of psychological evidence," *Intl. Journal of Imaging Systems and Technology - Facial Image Processing, Analysis, and Synthesis*, vol. 13, no. 1, pp. 3-7, 2003.

[26] G. Purcell and A. Stewart, "The face-detection effect," *Bulletin of Psychonomic Society*, vol. 24, pp. 118-120, 1986.

[27] J. Galbally, J. Fierrez, and J. Garcia, "Vulnerabilities in biometric systems: attacks and recent advances in liveness detection," *Database*, vol. 1, no. 3, pp. 1-8, 2007.

[28] R. Tronci et al., "Fusion of multiple clues for photo-attack detection in face recognition systems," in *Proc. Intl. Joint Conf. on Biometrics*, 2011.

[29] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971-987, 2002.

[30] J. Määttä, A. Hadid, and M. Pietikäinen, "Face spoofing detection from single images using micro-texture analysis," in *Proc. of International Joint Conference on Biometrics*, 2011.

[31] T. Pereira, A. Anjos, J. Martino, and S. Marcel, "LBP-TOP based countermeasure against facial spoofing attacks," in *Proc. of International Conference on Computer Vision*, vol. 1, pp. 121-132, 2012.

[32] S. Parveen et al., "Face liveness detection using Dynamic Local Ternary Pattern (DLTP)," *Computer*, vol. 5, no. 2, pp. 1-15, 2016.

[33] G. Souza, D. Santos, R. Pires, A. Marana, and J. Papa, "Efficient transfer learning for robust face spoofing detection," in *Proc. of Iberoamerican Congress on Pattern Recognition*, 2017.

[34] O. Lucena, A. Júnior, V. Moia, R. Souza, E. Valle, and R. Lotufo, "Transfer learning using Convolutional Neural Networks for face anti-spoofing," in *Proc. of Intl. Conf. on Image Anal. and Recognition*, 2018.

[35] D. Menotti, G. Chiachia, A. Pinto, W. Schwartz, H. Pedrini, and A. Falcão, "Deep representations for iris, face, and fingerprint spoofing detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 864-879, 2015.

[36] S. Hochreiter and J. Schmidhuber, "Long-short term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[37] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[38] J. Ba, G. Hinton, V. Mnih, J. Leibo, and C. Ionescu, "Using fast weights to attend to the recent past," *ArXiv preprint arXiv:1610.06258*, 2016.

[39] D. Santos, G. Souza, and A. Marana, "A 2D Deep Boltzmann Machine for robust and fast vehicle classification," in *Proc. of Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 155-162, 2017.

[40] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *ArXiv preprint arXiv:1502.03167v3*, 2015.

[41] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," in *Proc. of European Conference on Computer Vision*, pp. 504-517, 2010.

[42] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. of International Conference for Learning Representations*, 2015.

[43] Y. Jia et al., "Caffe: convolutional architecture for fast feature embedding," *ArXiv preprint arXiv:1408.5093v1*, 2014.

[44] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499-1503, 2016.

[45] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face antispoofing based on color texture analysis," in *Proc. of International Conference on Image Processing*, pp. 2636-2640, 2015.